Analog Design

Carsten Wulff, carsten@wulff.no

Status: 0.5

I. Checklist

There are roughly 3 phases of analog design.

- Specification
- Design
- Tapeout

The specification phase is where you think deeply through the design.

Can the design meet the key parameters you need? How will I verify the circuit? Do I know how to make the circuit?

These questions and more, are so common that most companies will have checklists that we use when we review the specification, design, and tapeout.

These checklists are closely guarded secrets, as the content contain significant amount of knowledge accumulated over numerous blunders, mistakes, failure to imagine, and physics teaching us a lesson.

The design phase is where we make the schematic, and simulate the schematic. We explore circuit architectures, fix problem corners, check our design over temperature, voltage, process corners (slow transistors, fast transistors, mismatch)

The tapeout phase is where we translate the schematic into layout, check the design rules (DRC), do layout versus schematic (LVS) and extract circuit parasitics to check layout parasitic effects (LPE). And, of course, simulate most things again.

I've made a checklist below for the most common questions that you need to ask yourself.

A. Specification

Item	Description	Yes	Action
Functional description	Have you described what the IP shall do?		
Key parameters	Have you updated your key parameters in the README		
Architecture	Have you described the circuit architecture? How should it work?		
Realism	Do you know how to do what you plan to do?		
Verification plan	Have you described exactly what you need to check? For example, stability of OTAs, current consumption, key parameters		
Specification	Have you added a specification for all parameters you intend to check. For example, phase margin should always be larger than 45 degrees.		

B. Design

Item	Description	Yes	Action
git	Have you committed the schematics to the repository?		
git push	Are the schematics pushed to github? Are you sure?		
git tag	Is the current version tagged		
Implementation	Are all schematics described with their own markdown file?		
Verification plan	Are all items on the verification plan completed? If not, have you described why it's no longer relevant?		
Electrical parameters	Are the electrical parameters updated with simulated results?		
Spec violations	Have you explained why the specification violations are not an issue?		
Simulation	Are the required corners run (typical, slow, fast, mc)		

C. Tapeout

Item	Description	Yes	Action
git	Are all schematics and layout committed? Are you sure?		
git push	Have you pushed to github?		
git tag	Is the latest version tagged?		
LVS	Is the LVS on github passing (green)?		
DRC	Is the DRC on github passing (green)?		
LPE	Is the LPE on github passing (green)?		
Simulation	Are the required corners re-run with layout parasitics (typical, slow, fast, mc)		

II. Schematic rules

Rules are nice. They reduce the cognitive load since a decision already has been made. You may disagree with the rule, but in analog design it's not that important what the "rule" is, but sometimes more important that the rule is followed.

Naming rules are one example. We can have a philosophical discussion till the end of time of whether it's best with uppercase or lower case. CamelCase, however, is wrong in schematics and SPICE, since SPICE is case-insensitive, so "vref" is the same as "Vref**.

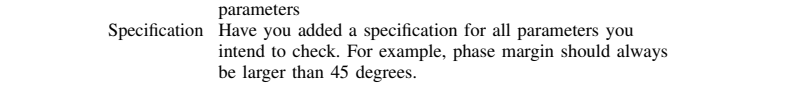
Below are the rules I like. You may disagree, but if you're my student, then you don't have a choice. Follow them, or the grade may suffer.

1) Only uppercase names allowed: **Do:** AVDD, **Don't:** aVdd

Although editors can handle mix of upper case and lower case, SPICE, cannot. SPICE is case insensitive. That means AVDD == aVdd in SPICE, but AVDD != aVdd in editors. A such mixing case is a bad idea, so one must be picked, and uppercase the chosen one. Why? Why not?

2) Use same net throughout hierarchy: **Do:** VDD -> VDD -> VDD, **Don't:** VDD -> LOCALVDD -> CELLVDD

Debugging becomes a lot simpler if a net keeps it's name throughout the schematic hierarchy. Especially bad are cases where a net name is reused on multiple levels of hierarchy. Imagine the following scenario



| |

Here the net name VDD is used on the top level, while LVDD is used in the sub-block. In the sub-block there is a power switch between LVDD and VDD. In this case VDD != VDD on top level, which can lead to long debugging times. There are, however, a few exceptions. For example, using VDD on standard cells (inverters, ANDs etc) is ok, even though the power supply is not called VDD

3) Spend time on making schematics pretty: It matters how schematics look. Think of it like this. In 10 years, you will be asked to port, re-simulate, fix a bug, on your design. If you have spent some time adding comments, making things look pretty, etc, then the job will be much, much easier. "A pretty schematic is a love letter to your future self".

> PS: This applies to documentation, code, and everything you make.

4) All digital nets must be "active high"" naming: **Example:** PWRUP_3V3, PWRUP_N_3V3

The PWRUP_3V3 should be understood as "When PWRUP_3V3 is high, then the block is powered up"

The PWRUP_N_3V3 should be understood as "When PWRUP_N_3V3 is high then the block is powered down"

5) Bias currents shall be named IB<device sending current (P|N)>: **Do:** IBP_1U, **Don't:** IBIAS_1U

The "P" post-fix tells us the current comes from a PMOS, so we can put it into a NMOS diode connected transistor. On the IBIAS we have no idea which way the current flows.

III. Layout rules

1) Limit the amount of transistor Width's and Length's that you use: **Do:** W = 1.0 um, L = 180 nm, Use multiplier for other sizes

Don't: W1 = 1 um, W2 = 1.1 um, W3 = 1.2 um, W4 = 2 um, W5 = 2.1 um

You want the layout to be relatively regulator. A bunch of different Ls and Ws is a pain when you do layout

2) Use pre-defined transistors for regular layout: **Do:** Use JNW_ATR_SKY130A and JNW_TR_SKY130A

3) Always use two fingers for analog transistors: That way, you don't have to worry about current direction in the layout.

4) Always run gates in the same direction: Mobility of transistors (especially PMOS) is affected by strain, so if you rotate a transistor it will not have the same current, and change in current as a function of stress.

I've seen ICs have to be taped out again due to rotated transistors.

5) Always have dummy poly gates: For large lengths (> 500 nm) the lithography effects are not that severe, but the etching of the gate material will be asymmetric if there are no poly dummies. Make sure the poly dummy is exactly the same spacing on both sides.

For small lengths (< 200 nm) the lithography effects start to matter. The light used in most litho is 193 nm. 193 nm is used all the way down to about 7 nm.

Due to diffraction effects, it's common to have extremely regular poly spacing, an exact distance such that the interference from neighboring poly's align perfectly to the next poly.

6) Always place transistors away from well edge: Close to the N-well edge the donor consentration will be higher. Ion implantation is used for the wells, and the ions will scatter of the oxide wall, and increase the doping concentration close to the edge. As such, the transistor threshold voltage will increase close to a well edge.

Keep transistors about 3 um away from the N-well edge if threshold voltage is important.

Carsten Wulff received the M.Sc. and Ph.D. degrees in electrical engineering from the Department of Electronics and Telecommunication, Norwegian University of Science and Technology (NTNU), in 2002 and 2008, respectively. During his Ph.D. work at NTNU, he worked on open-loop sigma-delta modulators and analog-to-digital converters in nanoscale CMOS technologies. In 2006-2007, he was a Visiting Researcher with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada. Since 2008 he's been with Nordic Semiconductor in various roles, from analog designer, to Wireless Group Manager, to currently Principle IC Scientist. From 2014-2017 he did a part time Post.Doc focusing on compiled, ultra low power, SAR ADCs in nanoscale technologies. He's also an Adjunct Associate Professor at NTNU. His present research interests includes analog and mixed-signal CMOS design, design of high-efficiency analog-to-digital converters and low-power wireless transceivers. He is the developer of Custom IC Compiler, a general purpose integrated circuit compiler, and makes the occational video on analog integrated circuits at https://www.youtube.com/@analogicus. For full CV see https://analogicus.com/markdown-cv/.